### Memberbl模块

1. 模块概述

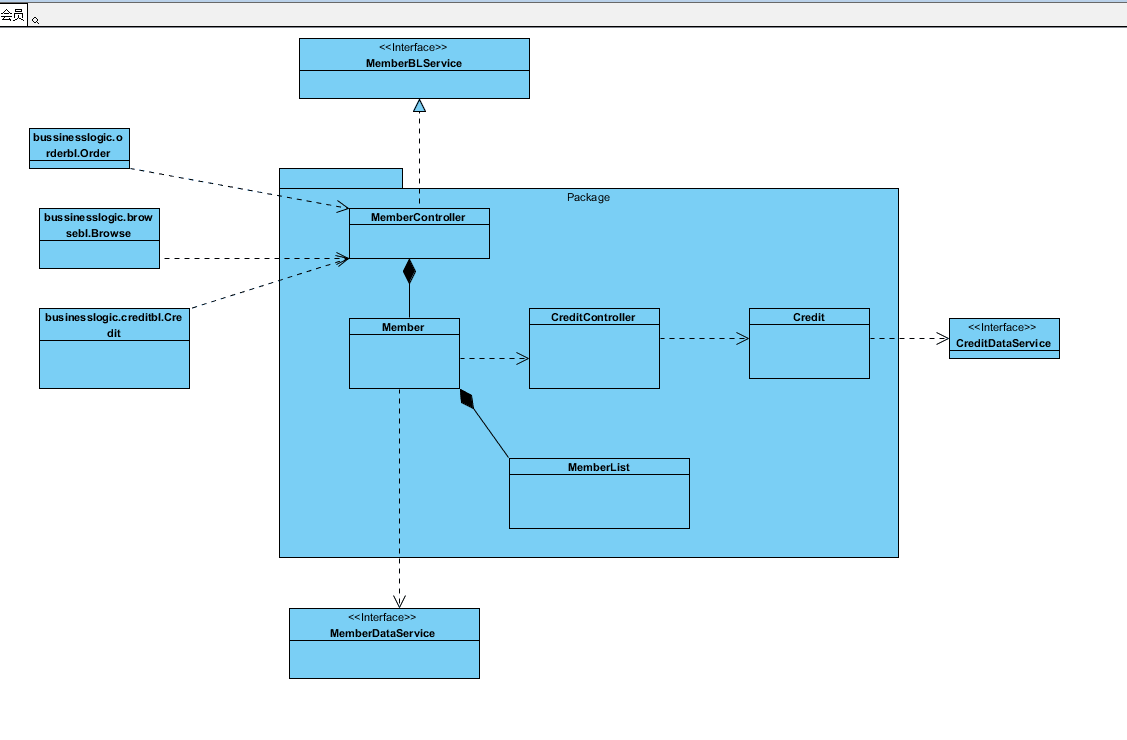
memberbl模块承担的需求参见需求规格说明文档中注册会员相关内容及相关非功能需求

memberbl模块承担的职责和接口见体系设计文档中的memberbl相关的职责和接口

1. 整体结构

依先前体系设计逻辑模型和相关文档的说明，为了增加灵活性降低耦合，增加可复用性，在界面层和业务逻辑层增加接口层，在数据层和业务逻辑层也增加接口层，通过接口进行调用，其中，使用membercontroller将模块里的业务逻辑和逻辑操作相分开，将注册删除及获取会员信息委托给member.

memberbl 模块的设计如下图所示



memberbl模块各个类的职责由下表所示

|  |  |
| --- | --- |
| 类 | 职责 |
| MemberController | 负责实现有关会员权利的服务 |
| Member | 会员的领域模型对象，拥有一个会员的各种属性（包括信用值、ID、联系方式、企业名称、生日、会员标记）以及注册VIP、获得会员信息、更改会员信息的方法，可以帮助完成会员界面所需要的服务 |

1. 模块内部类的接口规范

MemberController、Member的接口规范如下表所示

MemberController:

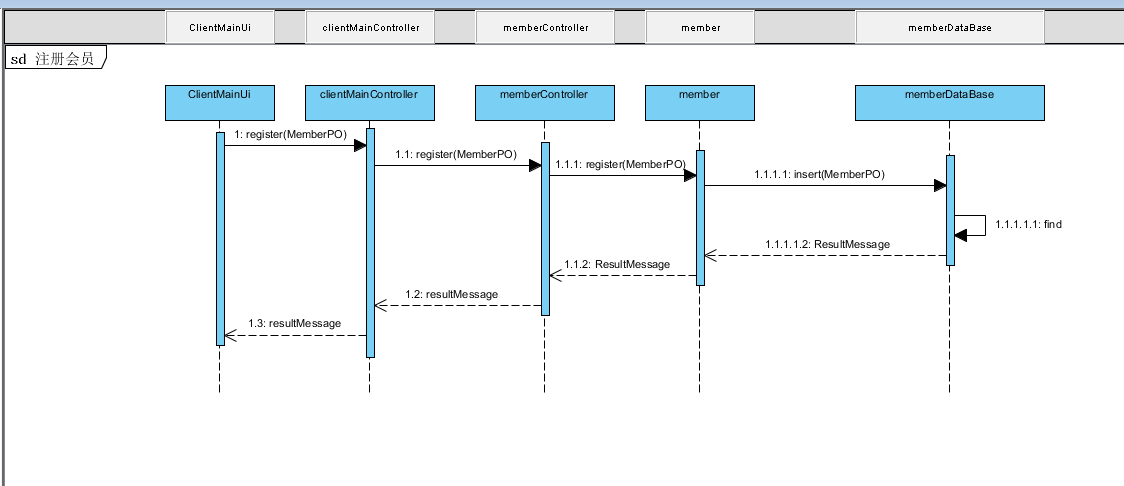
|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| MemberController.register | 语法 | Public ResultMessage register(VIPVO vo); |
| 前置条件 | 已创建一个member领域对象，并且输入符合输入规则 |
| 后置条件 | 调用member领域对象的register方法 |
| MemberController.getinfo | 语法 | Public MemberPO getInfo(long Userid); |
| 前置条件 | 已创建一个member领域对象 |
| 后置条件 | 调用member领域对象的getinfo方法 |
| MemberController.cancel | 语法 | Public ResultMessage cancel(long Userid); |
| 前置条件 | 已创建一个member领域对象 |
| 后置条件 | 调用member领域对象的cancel方法 |
| MemberController.changeinfo | 语法 | Public ResultMessage changeInfo(MemberVO) |
| 前置条件 | 已创建一个member领域对象，并且输入符合规则 |
| 后置对象 | 调用member领域对象的changeInfo方法 |
| MemberController.manageInfo | 语法 | Public memberlist manageInfo() |
| 前置条件 | 已创建一个member领域对象 |
| 后置条件 | 调用member领域对象的manageInfo方法 |
| MemberController.delete | 语法 | Public ResultMessage delete(long userid) |
| 前置条件 | 已创建一个member领域对象 |
| 后置条件 | 调用member领域对象的delete方法 |
| 需要的接口 | | |
| 服务名 | 服务 | |
| Member.register(String phonenumber,String companyname,Time time) | 创建一个会员VIP对象 | |
| Member.getinfo(long Userid) | 返回一个会员的详细信息（包括会员ID、联系方式、用户名、企业名称、生日、信用值） | |
| Member.cancel(long Userid) | 删除一个会员对象 | |
| Member.changeInfo(MmeberVO) | 更改会员的基本信息 | |
| Member.manageInfo() | 返回所有用户信息 | |
| Member.delete(long userid) | 在用户管理时删除一个用户信息 | |

Member:

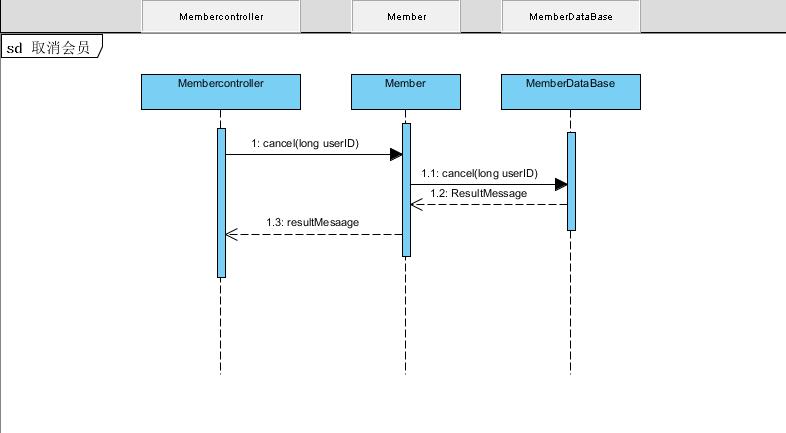
|  |  |  |
| --- | --- | --- |
| 提供的服务 | | |
| Member.register | 语法 | Public ResultMessage register(VIPVO vo); |
| 前置条件 | 启动一个会员注册回合 |
| 后置条件 | 在一个会员注册回合中，增加一个会员信息 |
| Member.getinfo | 语法 | Public Memberinfo getInfo(long Userid); |
| 前置条件 | 已经完成会员注册 |
| 后置条件 | 返回根据Id相匹配的会员信息 |
| Member.cancel | 语法 | Public ResultMessage cancel(long Userid); |
| 前置条件 | 已经完成会员注册且当前信用值不足标准信用值 |
| 后置条件 | 删除与id相匹配的会员信息 |
| Member.changeInfo | 语法 | Public ResultMessage changeInfo(MemberVO) |
| 前置条件 | 启动一个会员修改信息回合 |
| 后置条件 | 修改会员的信息 |
| Member.manageInfo | 语法 | Public Memberlist manageInfo() |
| 前置条件 | 启动一个用户管理回合 |
| 后置条件 | 查看数据库中的会员信息 |
| Member.delete | 语法 | Public ResultMessage (long userid) |
| 前置条件 | 启动一个用户管理回合 |
| 后置条件 | 删除数据库中一个用户的信息 |
| 需要的接口 | | |
| 服务名 | 服务 | |
| MemberDataService.insert(MemberPO po) | 插入单一持久化对象 | |
| MemberDataService.getinfo(long Userid) | 返回一个单一持久化对象的信息 | |
| MemberDataService.cancel(long Userid) | 删除一个会员对象/用户对象 | |
| MemverDataService.Changeinfo(MemberPO po) | 更新单一持久化对象 | |
| DatabaseFactory.getMemberDatabase | 得到Member数据库服务的引用 | |

1. 业务逻辑层的动态模型

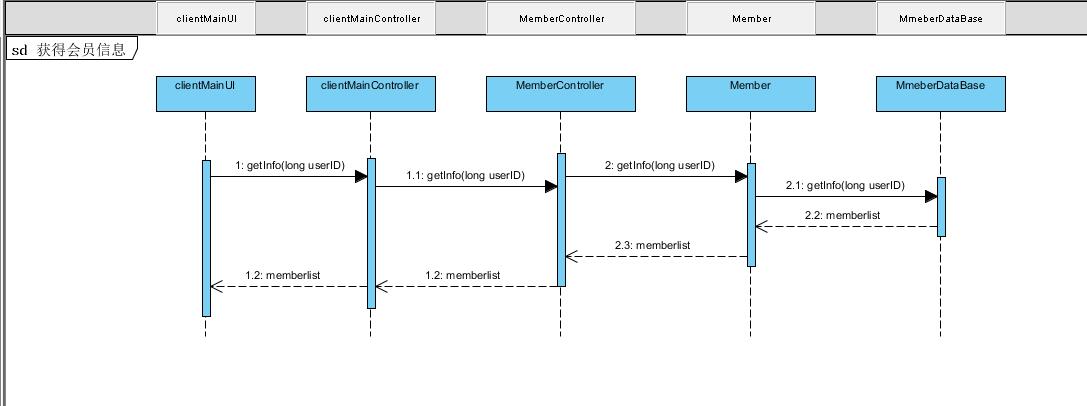
下图显示当用户请求注册会员后，Member业务逻辑层处理的各个对象之间的协作：



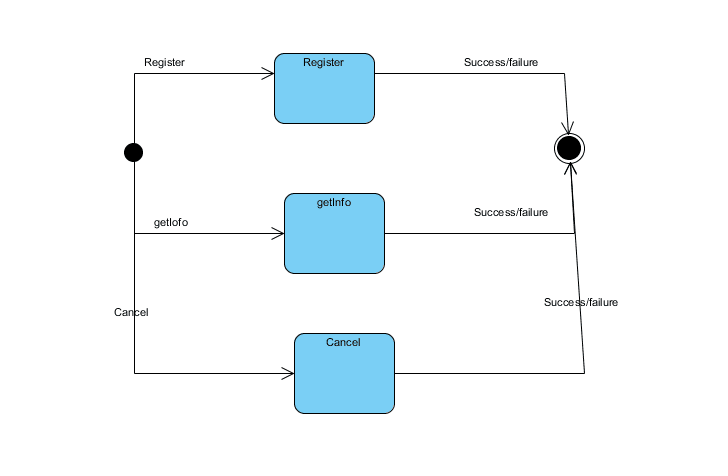
下图为用户信用值不足之后，Member业务逻辑层处理各个对象之间的协作：



下图为用户请求获取会员详细信息之后，Member业务逻辑层处理各个对象之间的协作：



下图描述了Mmeber对象的状态图，描述了对象的生存期间的状态序列，引起转移的时间，以及相伴随的操作，当有用户执行注册或查看个人详细信息，状态会进行验证并返回结果：



1. 业务逻辑层设计原理

采用委托式控制器风格，由controller统一调度，将任务委托给对应的对象进行处理。

### Commentbl模块

1. 模块概述

commentBl模块承担的需求参见需求规格说明文档中评价等相关隐含或非隐含的内容及相关非功能需求

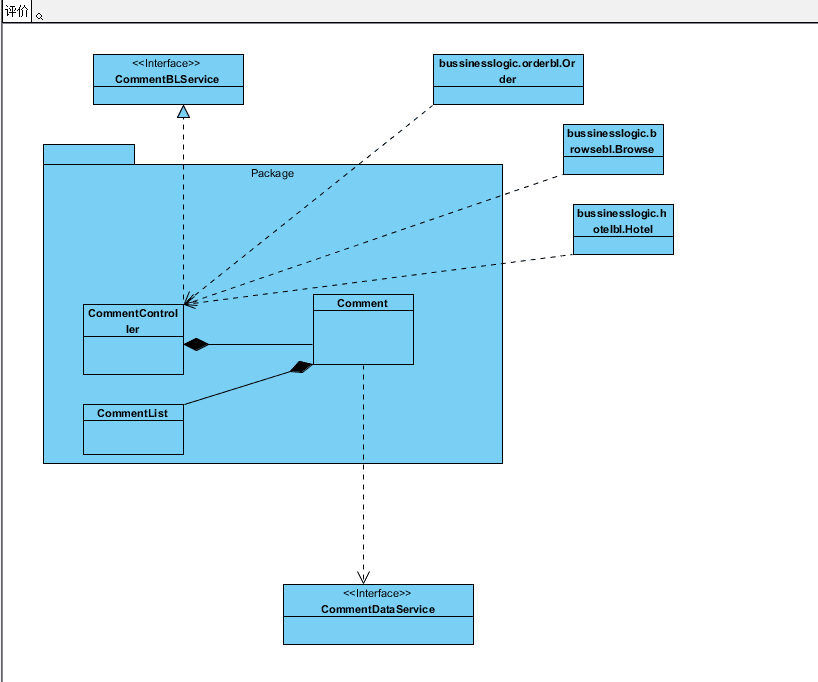
commentBl模块承担的职责和接口见体系设计文档中的commentBl相关的职责和接口

1. 整体结构

依先前体系设计逻辑模型和相关文档的说明，为了增加灵活性降低耦合，增加可复用性，在界面层和业务逻辑层增加接口层，在数据层和业务逻辑层也增加接口层，通过接口进行调用，其中，使用commentcontroller将模块里的业务逻辑和逻辑操作相分开，将评价及获取评价委托给comment，同时在登陆成功后调用相关member的接口进行初始化。

具体总体概述见userbl模块的整体结构前半段描述，这里只介绍内部具体，使用BrowseController将模块里的业务逻辑和逻辑操作相分开，直接控制Browse（这里Browse是一个基类），以适配器的风格，将浏览订单信息委托给BrowseOrderList，将浏览策略信息委托给BrowseStrategyList,各具体类型的适配器分别调用其他模块接口完成自己的工作。具体的返回也有相应的基本列表包装！

Commentbl 模块设计如下：



commentbl各个类的职责如下

|  |  |
| --- | --- |
| 类 | 职责 |
| commentController | 负责commentbl模块的逻辑操作，实现客户对订单评价及酒店管理人员对订单回评所需要的服务 |
| comment | 评价的领域模型对象，拥有一个订单评价对应的评分评价和文字评价以及提供进行评价和获取评价的方法，可以帮助完成评价界面所需要的服务 |

1. 模块内部类的接口规范

commentController、comment的接口规范如下表所示

commentController:

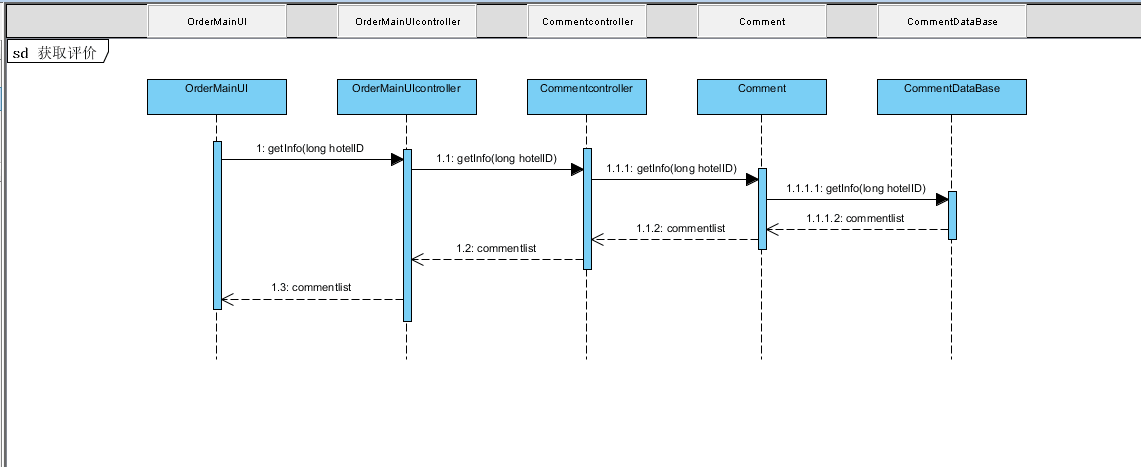
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| commentController.getinfo | 语法 | Public Commentlist gethotleInfo(long hotelid); |
| 前置条件 | 已创建一个comment领域对象 |
| 后置条件 | 调用comment领域对象的getinfo方法 |
| commentController.review | 语法 | Public ResultMessage review(CommentVO vo); |
| 前置条件 | 已创建一个comment领域对象，并且输入符合输入规则 |
| 后置条件 | 调用comment领域对象的review方法 |
| 需要的服务(需接口) | | |
| Comment.getinfo(long hotelid) | 得到基于rule的酒店信息列表 | |
| Comment.review(commentVO vo) | 得到hotel的详细信息 | |

comment类的接口规范如下：

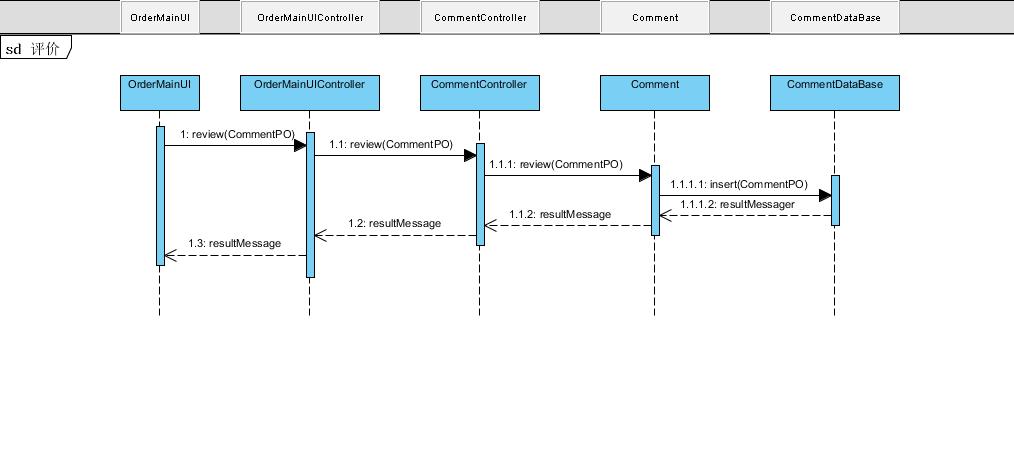
|  |  |  |
| --- | --- | --- |
| 提供的服务(供接口) | | |
| comment.getinfo | 语法 | Public Commentlist getinfo(long hotelid); |
| 前置条件 | 用户已登录，且这里用户为客户 |
| 后置条件 | 得到对应Hotelid的用户评价 |
| Comment.review | 语法 | Public ResultMessage review(CommentVO vo); |
| 前置条件 | 用户已经登录并启动一个评论回合 |
| 后置条件 | 添加一个commentVO到comment数据库 |
| 需要的服务(需接口) | | |
| DatabaseFactory.getCommentDatabase | 得到Comment数据库服务的引用 | |
| CommentDataService.insert(CommentPO po) | 插入单一持久化对象 | |
| CommentDataService.getInfo(long hotelID) | 得到一个单一持久化对象信息 | |

(4)业务逻辑层的动态模型

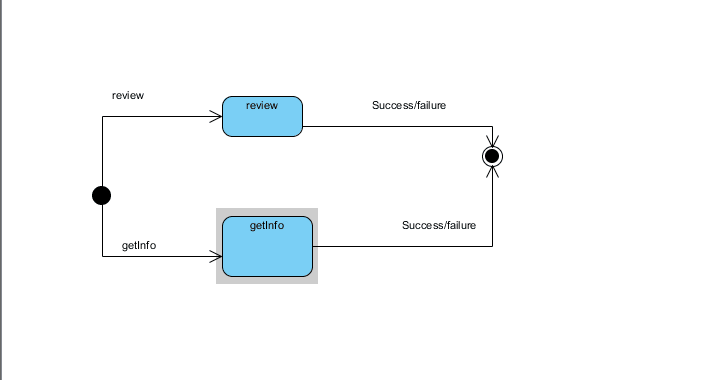
下图显示当用户请求查看评价时，Commentbl模块各个类之间的协作时序图



下图显示当用户请求对订单进行评价时，Commentbl模块各个类之间的协作时序图



下图描述了comment对象的状态图，描述了对象的生存期间的状态序列，引起转移的时间，以及相伴随的操作。当客户发起了一个请求，comment处于抽象状态，控制器解析后调用相应方法（并声明了具体类型），comment变为具体状态并处理请求，处理完后结束



（5）业务逻辑层设计原理

采用适配器模式风格，由controller统一调度，将任务委托给comment，comment对类型进行转换后处理结束。